Zusammenfassung DASB

Data Science Basics

Maurin D. Thalmann

24. Januar 2020

Inhaltsverzeichnis

1 Learning Objectives

Introduction to Data Science

- Define the term „Data Science“ and differentiate it from similar concepts such as big data, data mining, OLAP, predictive modeling, data analysis
- Describe why and how value is created from data, especially in form of processes
- Describe the skills and activities of a data science professional
- Describe possible applications of data science

Intro to R and Exploratory Data Analysis (EDA)

- Define the term „Exploratory Data Analysis“ and explain why it is a fundamental part of making data science
- Give a short introduction to R and present its main features
- Indicate the key actors and actions that compose EDA and how they are related to the data sources considered
- Indicate different ways of importing data into R for analysis and how to check for their quality
- Describe the main operations available in R (through libraries) to manipulate data and prepare it for the analysis

ggplot2 and its usage in EDA

- Define the role of graphical representations in the „Exploratory Data Analysis“ and explain why it is a fundamental tool to make it effective
- Give a short introduction to ggplot2 and its features
- Ddescribe the main way of working of ggplot2 and how to compose complex graphs (e.g.: representing more than 3 variables in a single graph, for multidimensional data)
- Explain the role of the facets and how they can support the representation of „partial“ relationships
- Indicate the role of the coordinates in the graph usage for EDA

Intro to Forecasting

- Understand and explain the concept of forecasting
- Present the basic approach, either using the theory or by example
- Present a linear model and describe its basic characteristics
- Indicate how EDA can support and scaffold the model identification
- Apply basic forecasting in R, given the datasource

Forecasting Models: Usage / Application

- Presents the limits of the linear models
- Define the way of extending linear models
- Justify forecasting & interpretation (phenomena comprehension) in data science, also by presenting examples
- Present advantages and limits of the defined extensions
- Presenting Parametric and Nonparametric idea in model definition (ML)
- Indicate how EDA can support and scaffold the model identification
- Apply more advanced forecasting in R, given the datasource

2 Introduction to Data Science

2.1 Introductory Stizzle

- More is different - with different quantity comes different quality, although „more“ is always dependent on the point of view.
- Data volume nowadays is exploding (276.12 billion GB of digital data)
- Data Science tries to close the gap between „Data *available* to an organization“ and „Percent of data an organization can *process*“.

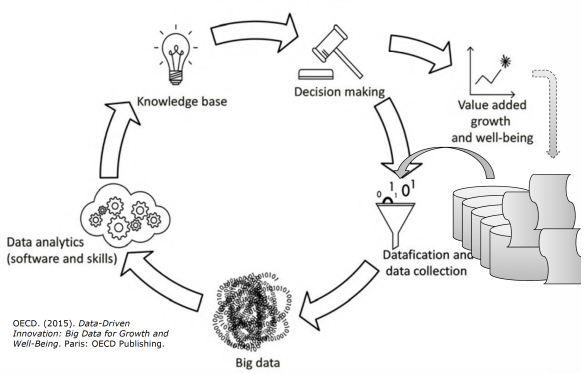


Abbildung 1: Beautiful illustration of the classic data value cycle

„**Data Science** is the extraction of actionable knowledge directly from data through a discovery, or hypothesis formulation and hypothesis testing“. Data Scientists generate knowledge from big data.

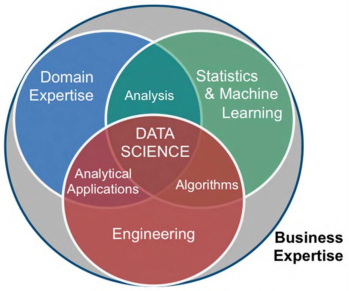


Abbildung 2: Skills needed in Data Science

2.2 A Global View on Data Science

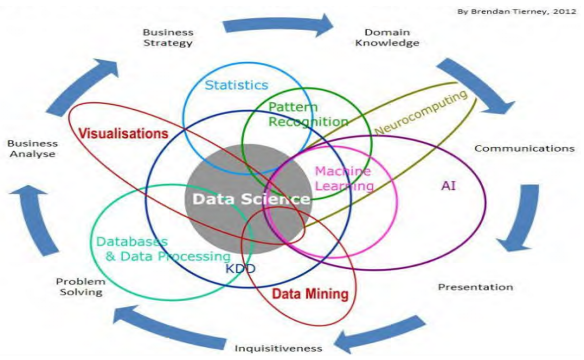


Abbildung 3: Data Science is multidisciplinary

- **AI**: programs that perform tasks resembling humans (learn and reason)
- **ML**: algorithms to learn from that data without explicit programming
- **DL**: subset of ML using artificial neural networks for treating vast amount of data (big data)
- **Data Science**: spans the collection, management, analysis and interpretation of large amounts of data with a wide range of applications → make informed decisions based on what was learned
- **EDA** (exploratory data analysis): extract insight from data (outside AI, human based)

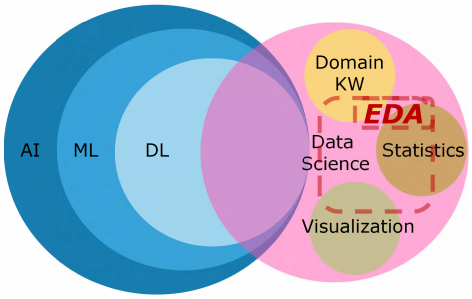
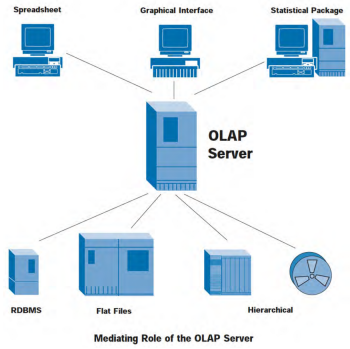


Abbildung 4: AI / ML / DL and Data Science

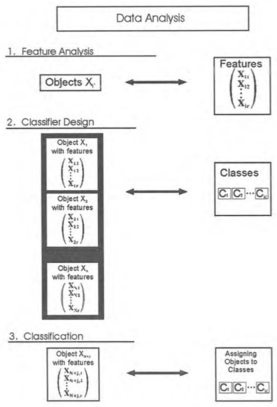
2.3 Different Concepts in Data Analytics

2.3.1 1993 - Online Analytical Processing (OLAP)



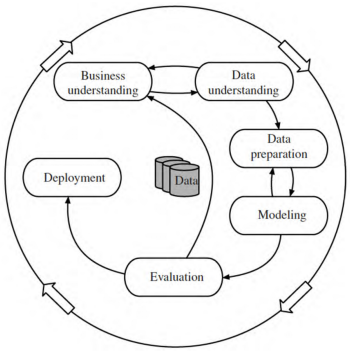
- It's online, not batch (interactive, not programmed in COBOL)
- The OLAP system should access the data required to perform the indicated analysis
- OLAP tools empower useranalysts to earily perform multi-dimensional analysis, which previously have been avoided because of their perceived complexity.

2.3.2 1998 - Fuzzy Data Analysis



- Data analysis can be defined as **search for structure** in data
- In data analysis, objects are considered which are described by some attributes
- Most of the traditional methods for data analysis assume that patterns to be detected are two-valued
- Whenever this is not the case, the relationship between data and classes becomes gradual
  → Fuzzy Classification

2.3.3 2005 - Data Mining



- Very similar concept to data science
- Machine Learning (modeling) is the technical core of practical data mining applications
- Data *Mining* is a business process related to *value* (finding the metaphorical gold nugget)
- The lifecycle of a data mining project is defined by the CRISP-DM reference model

2.3.4 2013 - Predictive Modeling

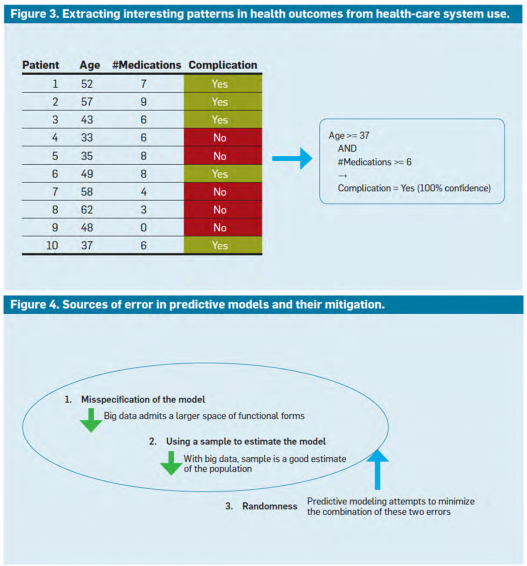


Figure 3. Extracting interesting patterns in health outcomes from health-care system use.

Patient	Age	#Medications	Complication
1	52	7	Yes
2	57	9	Yes
3	43	6	Yes
4	33	6	No
5	35	8	No
6	49	8	Yes
7	58	4	No
8	62	3	No
9	48	0	No
10	37	6	Yes

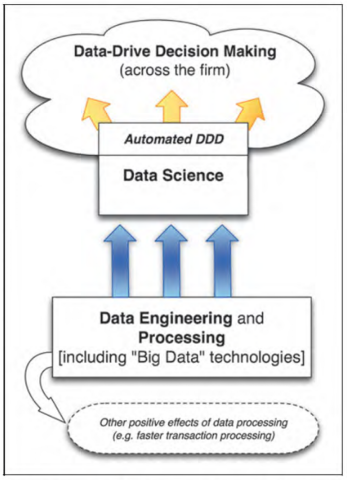
Age >= 37
AND
#Medications >= 6
→
Complication = Yes (100% confidence)

Figure 4. Sources of error in predictive models and their mitigation.

1. Misspecification of the model
Big data admits a larger space of functional forms
2. Using a sample to estimate the model
With big data, sample is a good estimate of the population
3. Randomness
Predictive modeling attempts to minimize the combination of these two errors

- A common epistemic requirement in assessing whether new knowledge is actionable for decision making in its *predictive power*, not just its ability to explain the past.
- The requirement on predictive accuracy on observations that will occur in the future is a key consideration in data science.

2.3.5 2013 - Data Driven Decision Making



- Data science involves much more than just data-mining algorithms.
- Successful data scientists must be able to view business problems form a *data perspective*.

3 Intro to R and Exploratory Data Analysis (EDA)

3.1 Intro to R

- **R** ist a software and programming language for statistical applications and graphics
- It's an implementation of the programming language S, with some extensions
- It provides functions for linear/non-linear modelling, classical statistical tests, time-series analysis, classification, clustering etc.
- **RStudio** is an integrated development environment (IDE) for R

3.1.1 Basic Operation in R

- Assignment:
  `my_var <- 46, nr <- 38`
  (You can also use `copy my_var = 46`, but it is not suggested)
- Printing (explicit and implicit): every line with only a name of an object
  Implicit: `my_var`
  Explicit using functions like print: `print(my_var)`
- Arithmetic operators:
  `4 + 5`
  `2 ^ 3 - 7 %% 3 - 2 ^ (3 - 7 %% 3)` etc. etc. etc.
- Control flow, conditionals, loops can be used as in any other language
  `for (i in 1:10) print(i)`
  `while (nr < 42) print(,,We have ‘‘+str(nr)+,, : still space...‘‘); nr=nr+1`

3.2 What is EDA

- Exploratory Data Analysis is the
  - critical process of performing initial investigations on data so as to
    - discover patterns
    - spot anomalies
    - test hypothesis
    - check assumptions
  - with the help of **summary statistics** and **graphical representations**

The use of a statistical model is not mandatory, in this step, even if it is normally adopted to test hypotheses and check assumptions. For anomalies and patterns discovery, correlations (represented into the correlation matrix) and distribution (represented with box and whisker diagrams) are the usual approaches.

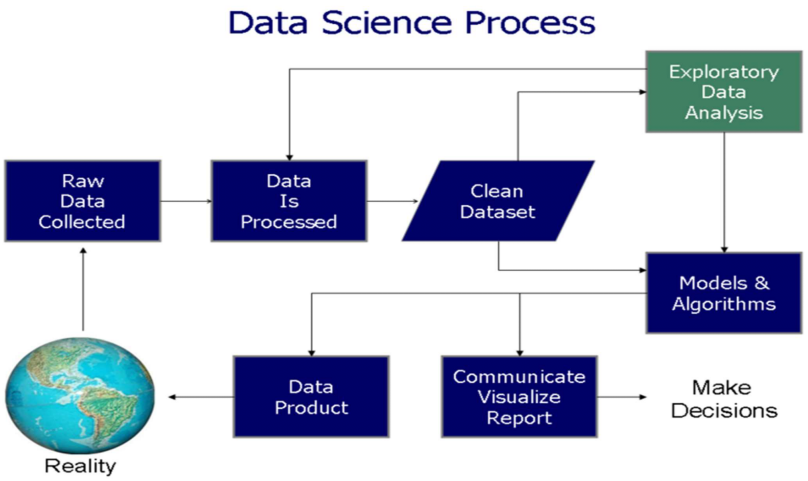


Abbildung 5: EDA in the context of Data Science

3.2.1 EDA process „in practice“

- EDA is an iterative cycle
- Process:
  - Generate questions about the data
  - Search for answers by visualizing, transforming and modelling the data
  - Use the insights to refine the questions and/or generate new questions
- Start with many ideas...
- some will be evidently useless and other will end up in a dead end
- a few of them will help to have a better understanding of the data/process
- repeat with generating other ideas until the insight is enough
- then you can apply the model to the data for forecasting and process explanation
- EDA also allows you to verify/test the quality of your data

3.3 Data Import / Transform / Link

3.3.1 Data Import

- R provides importing functions, depending on the format of the data source:
  - CSV / TSV / delimited texts: `read_csv()` & `read_csv2()` (semicolon), `read_tsv()`, `read_delim()`
  - Fixed width columns: `read_fwf()` & `read_table()`
  - Logs (e.g. webserver logs): `read_log()`

These are all similar in terms of parameters, examples:

```
read_delim(file, delim, quote = "\"", escape_backslash = FALSE, escape_double = TRUE,
col_names = TRUE, col_types = NULL, locale = default_locale(), na = c("", "NA"),
quoted_na = TRUE, comment = "", trim_ws = FALSE, skip = 0, n_max = Inf, guess_max =
```

3.3.2 Data Transform (example dataset)

Example dataset loaded from: *Auto MPG Data Set*

```
> library(tidyverse)
> DS <- read.table("C:/Users/JumpStart/Downloads/auto-mpg.data",
quote="\"", comment.char="")
> names(DS) <-
c("mpg","cylinders","displacement","horsepower","weight","acceleration","
model year","origin","car name")
> DS <- as_tibble(DS); attach(DS); View(DS)
```

3.3.3 Data Transform

- Filter rows with `filter()`
  - subset observations based on their values
    `filter(DS, weight > 3500)` or `filter(DS, DS$weight > 3500)`
- Arrange rows with `arrange()`
  - changes the order of rows
    `arrange(DS, desc(DS$acceleration))`
- Select columns with `select()`
  - Subset columns with position and/or negative condition (by name or index)
    `select(DS, -1)`
    `select(DS, 2, 3, 4)`
    `select(DS, c(mpg, 5))`
- Add new variables with `mutate()`
  - Allow to create new dimension(s)
    `mutate(DS, a4w = acceleration/weight, pre75 = as.numeric(DS$'model year' < 75))`
- Grouped summaries with `summarise()`
  - Allows to compute aggregate metrics
    `summarise(group_by(DS, 'model year'), mean_display = mean(displacement), sd_displ = sd(displacement), n=n()`

- All verbs work similarly:
  - The first argument is a dataframe
  - The subsequent arguments describe what to do with the data frame, using the variable names (without quotes)
  - The result is a new data frame

3.3.4 Data „Linkage“

Some prices of cars of the 70's: *70's Cars*.
This data is imported into a new tibble and we will try to use it for creating another View for exploring the relationship between cost, year, power and mpg.

```
> car_cost <-read_delim("C:/Users/JumpStart/switchdrive/__/DASB/SW02/car_197x_costs_2.csv",
    delim = "-",
    escape_double = FALSE,
    na = "NA",
    trim_ws = TRUE,
    col_types = cols(`Car Name` = col_factor(),
    Price = col_character(),
    Location = col_character(),
    `Matriculation Year` = col_integer()))
> car_cost <- as_tibble(car_cost)
> car_cost <- mutate(car_cost, Location = sapply(Location,as.factor))
> car_cost <- mutate(car_cost, Price = as.integer(str_replace(str_sub(Price, 2, -1),",","")))
> View(car_cost)
```

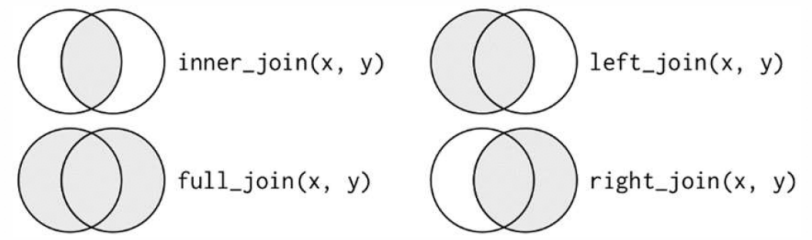


Abbildung 6: R functions to link data frames with each other

There are different types of R functions for linking tibbles/dataframes (~datasets)

- **Mutating** joins allows to combine variable from two tables
  - `inner_join(x, y, by = keys)`
    matches pairs of observations whenever their keys are equal
  - Outer join keeps observations that appear in at least one of the tables. There are three types of outer joins:
    - `left_join(x, y, by = keys)`
      keeps all observations in x
    - `right_join(x, y, by = keys)`
      keeps all observations in y
    - `full_join(x, y, by = keys)`
      keeps all observations in x and y
- **Filtering** joins allow to restrict the set of row from the 1^{st} table
  - `semi_join(x, y, by = keys)`
    keeps all observations in x that have a match in y
  - `anti_join(x, y, by = keys)`
    drops all observations in x that have a match in y
- **Set Operations** (work on complete rows, comparing each variable values. Inputs needs the same variables, for treating observations like sets)
  - `intersect(x, y)`
    return only observations in both x and y
  - `union(x, y)`
    return unique observations in x and y
  - `setdiff(x, y)`
    return observations in x, but not in y

3.3.5 Applying data loading/transforming/linking

- Find one or more data sources you are interested into (UCI ML Repository / Kaggle)
- Start the EDA „mindset“:
  1. What I believe about the data / what I would like to discover
     $\rightarrow$ Make hypotheses and formulate assumptions
  2. Think how this can be supported by data $\rightarrow$ experiment
  3. Run the experiment to verify
  4. Acquire new knowledge that can help you improve step 1
- Repeat the process at least 3 times
  (Optional: short report of the process you followed, document it in a short presentation)

4 ggplot2 and its usage in EDA

4.1 ggplot2 - Intro

- Used with the library „ggplot2“
  `library(ggplot2)`
- Simple usage example:
  `plot <- ggplot(data = mpg, mapping = aes(x = mpg$hwy))`
  plot + geom_histogram()
- ggplot 2 package:
  - produces layered statistical graphics.
  - uses an underlying „grammar“ to build graphs layer-by-layer rather than providing premade graphs
  - is easy enough to use without any exposure to the underlying grammar, but is even easier to use once you know the grammar
  - allows the user to build a graph from concepts rather than recall of commands and options

4.2 ggplot2 - layer after layer

- The initial call to ggplot does create the graph, but no content:
```
ggplot(data = mpg)
ggplot(data = mpg, mapping = aes(x = mpg$hwy))
ggplot(data = mpg, mapping = aes(x = mpg$hwy, y = mpg$dipl))
```
- We need to add a layer by using + (or collect the partial graph in an object)
```
p <- ggplot(data = mpg)
p + geom_histogram(mapping = aes(x = mpg$hwy))
```
- Parameters set into the ggplot() call are inherited by every other layer, but can be overridden, if needed
```
p <- ggplot(data = mpg)
p <- p + geom_histogram(mapping = aees(x = mpg$hwy))
p
p + geom_violin(mapping = aes(y = mpg$displ))
```

4.3 Elements of grammar of graphics

- **Data**: variables mapped to aesthetic features of the graph.
- **Geoms**: objects / shapes on the graph.
- **Stats**: statistical transformations that summarize data (e.g. mean, confidence intervals).
- **Scales**: mappings of aesthetic values to data values. Legends and axes visualize scales.
- **Coordinate systems**: the plane on which data is mapped on the graphic.
- **Faceting**: splitting the data into subsets to create multiple variations of the same graph (paneling).

4.4 Aesthetics

- Visual properties of objects on the graph (depends on the geom*)
- Commonly used aesthetics:
  x positioning along x-axis
  y positioning along y-axis
  color color of objects; for 2D-objects: the color of the object's outline (compare to fill below)
  fill fill color of objects
  linetype how lines should be drawn (solid, dashed, dotted etc.)
  shape shape of markers in scatter plots
  size how large objects appear
  alpha transparency of objects (value from 0 [transparent] to 1 [opaque])

4.5 Geoms

- Geometric shapes produced for the plot.
  - `geom_bar()`: bars with bases on the x-axis
  - `geom_boxplot()`: boxes and whiskers
  - `geom_density()`: density plots
  - `geom_histogram()`: histogram
  - `geom_line()`: lines
  - `geom_point()`: points (scatterplot)
  - `geom_rug()`: rug plot (2D display with the two 1D marginal distributions)
  - `geom_smooth()`: smoothed conditional means
  - `geom_text()`: text

4.6 Stats & Scales

- **Stats** → statistically transform data, usually as some form of summary (e.g.: mean, standard deviation, confidence interval etc.). Each stat function is associated with a default geom, so no geom is required for shapes to be rendered.
- **Scales** → control how a plot maps data values to the visual values of an aesthetic
  - `scale_color_manual()`:
    define an arbitrary color scale, by scecifying each color manually
  - `scale_color_hue()`:
    define an evenly-spaced color scale, by specifying a range of hues and the number of colors on a scale
  - `scale_shape_manual()`
    define an arbitrary shape scale, by specifying each shape manually
- Scales can also be applied to axes:
  - `scale_x_continuous()`
    map continuous to visual values - `scale_x_discrete()`
  - `scale_x_date(labels = date_format(„%m/%d“), breaks = date_breaks(„2 weeks“))`:
    treat x values as dates - scale_x_datetime()
  - `scale_x_log10()`:
    plot x on log10 scale
  - `scale_x_reverse()`:
    reverse direction of x-axis
  - `scale_x_sqrt()`:
    plot x on square root scale

4.7 ggplot2 - Initial Example

```
ggplot(diamonds, aes(x=carat, y=price, color=cut)) +
  geom_point() +
  scale_color_manual(
  values=c("red", "yellow", "green", "blue", "violet")
) +
scale_y_continuous(
  breaks=c(0,2500,5000,7500,10000,12500,15000,17500),
  labels=c(0,2.5,5,7.5,10,12.5,15,17.5),
  name="price(thousands of dollars)"
)
```

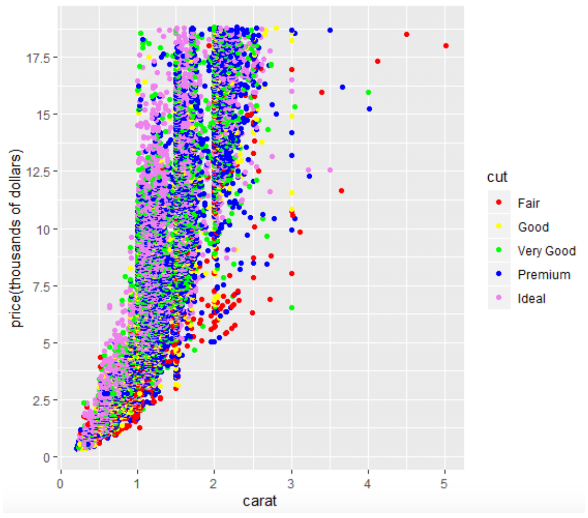


Abbildung 7: Rendered graph for Initial Example

4.8 Coordinate System & Faceting

- **Coordinate system**: set the planes for objects positioning on the plot
- Cartesian is the most diffused and used by default. Other coordinate systems would be:
  - polar
  - flipped Cartesian
  - fixed-ratio Cartesian
  - Map projections (mercatore, lagrange)
- **Faceting**: divide the graph into subgraphs (panels), using variable values
  - `facet_wrap(~{vars})`:
    create a multirow panel of plots, `{vars}` represents a list of splitting variables, separated by +
  - `facet_grid({vars}~{vars})`:
    row-splitting variable before, the column-splitting variable after
- Other aspects include the teming, position adjustment, labels and legends, zooming and the graph savings (ggsave)

4.9 ggplot2 - Second Example

```
ggplot(diamonds, aes(x=carat, y=price, color=cut)) +
  geom_point() +
  facet_grid(clarity~color) +
  scale_color_manual(values=c("red", "yellow", "green", "blue", "violet")) +
  scale_y_continuous(breaks=c(0,2500,5000,7500,10000,12500,15000,17500),
  labels=c(0,2.5,5,7.5,10,12.5,15,17.5),name="price(thousands of dollars)")
```

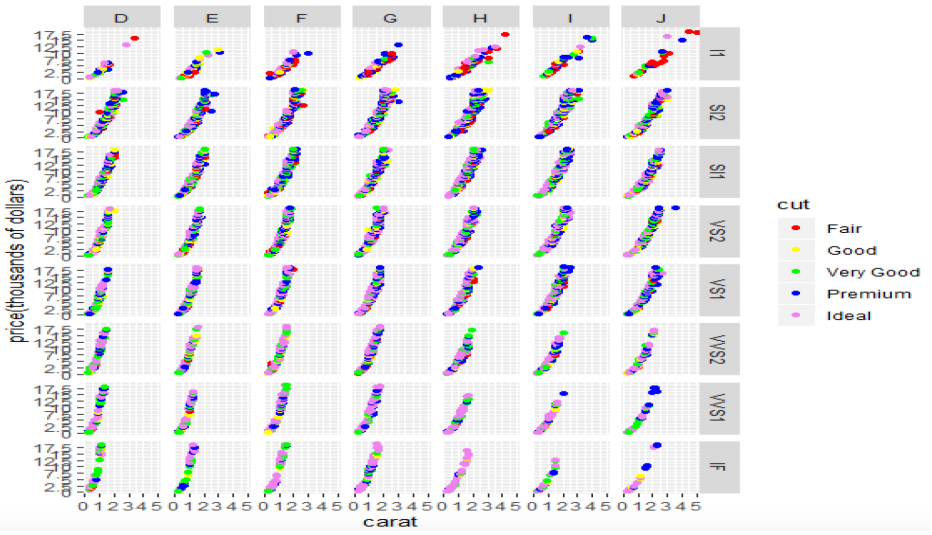


Abbildung 8: Rendered graph for Second Example

5 Shiny

Machi de wenni no Loscht han. (Wenn de Text am Schloss no do esch, hani kei Loscht meh gha)

6 Intro to Forecasting

6.1 Indication

- Forecast: the ability to use a set of variables for predicting another one
- Predictors $\rightarrow F(x) \rightarrow$ Output
- Can assume two forms:
  - Continuous Output $\rightarrow$ Regression
  - Discrete (Categorical) Output $\rightarrow$ Classification
- It uses similar approaches, but with a different result
- The function $F(x)$ is called the model. Can be used for two goals:
  - Pure prediction: care only for the forecast
  - Interpretation: care of the existing relationship, to better understand the phenomena

6.2 Linear Modeling

6.2.1 Modeling

- Simplest form of a model is the Linear Model
- With one predictor (bidimensional space) represent the equation of a line:
  $y = F(x) = \beta_0 + \beta_1 \cdot x$

$$Y \approx \beta_0 + \beta_1 \cdot X$$

$$e_i = y_i - \hat{y}_i$$

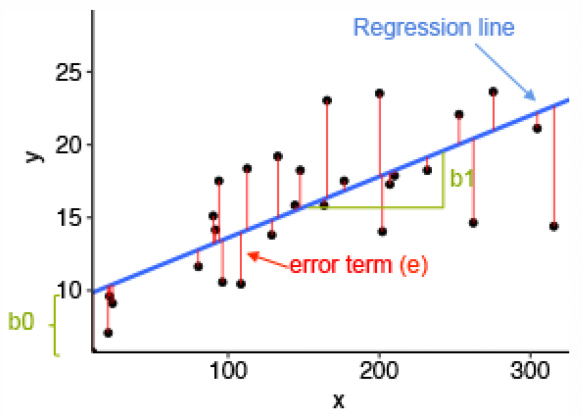
$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

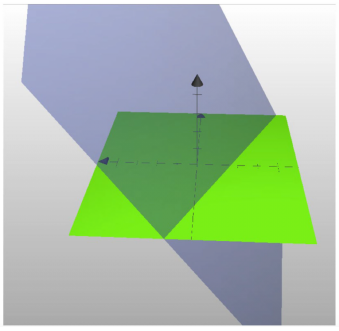
$$\bar{y} \equiv \frac{1}{n}\sum_{i=1}^{n} y_i$$

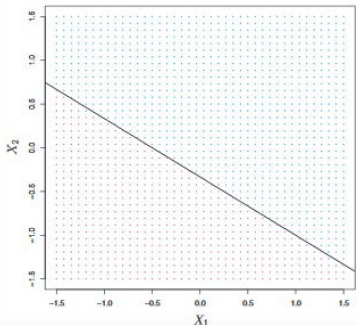
$$\bar{x} \equiv \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$\text{RSS} = e_1^2 + e_2^2 + \cdots + e_n^2$$



- With more predictors (multidimensional space or Hyperspace),
  the model can be seen as a hyperplane:
  $y = F(x) = \beta_0 + \beta_1 \cdot + \beta_2 \cdot x_2 + \cdots + \beta_n \cdot x_n$





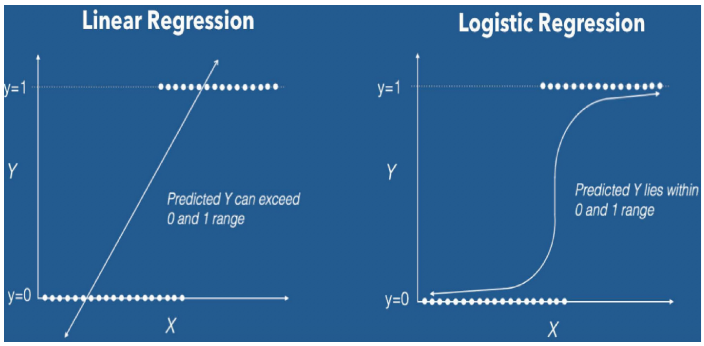
6.2.2 Model Identification

Following steps need to be done

- Define the predictors set and the output variables (also using correlation)
  - $x = \ldots, y = \ldots$
- Create the model
  - `model <- lm(y ~ x, data = d)`
- Inspect the model
  - Parameters $\beta_0, \beta_1, \beta_2, \ldots, \beta_n$
- Verify that the model is „trustable“
  - P-values of each β_a, everyone should be under 0.05 (5%)

6.2.3 Prediction

- It is natural to interpret the model for regression:
  - Given a single set of predictor values X (point) $\rightarrow$ compute the value of the output (y)
- but what about for classification?
  - We need to define a way to transform the output of the linear model to our output categories
  - This is done by the logistic function
- How to do predictions:
  - `Pred <- predict(model, data = d2)`



6.2.4 Model Evaluation

- We need to evaluate models based on performances
  - Regression $\rightarrow$ the sum of squared errors (RSS $\rightarrow$ residual sum of squares)
  - Classification $\rightarrow$ the misclassification error rate
- Every model with more predictors has a higher predictive power...
  ...but, we don't want to learn the noise on the data!
- Create an independent test set
  - Use most part (e.g. 70%) of the data for training purposes
  - Use the rest for the model performance estimation $\rightarrow$ Test Error Rate

7 Forecasting Models: Usage / Application

7.1 Forecasting vs. Interpretation

- Linear model
  - Rigid (strong assumptions about the relationship)
    - Predictors can only be represented by variables
    - Predictors do not have interactions (can be analyzed separately)
  - $y = F(X) = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$
- Two extensions:
  - Transformation applied to variables, to be used as predictor and/or output
    - E.g.: $\log_2(y) = F(X) = \beta_0 + \beta_1 \sin x_1 + \cdots + \beta_n e^{x_n}$
    - Formula $\rightarrow y \sim I(\sin(x_1)) + \cdots + I(\exp(x_n))$
  - Interaction between variables
    - E.g.: $y = F(X) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{1\_2} x_1 x_2$
    - Formula $\rightarrow y \sim x_1 \cdot x_2$

7.2 Performance vs. Interpretability

- Every extension adds more predictive performances, but reduces interpretability.
- Limits (boxes between Input $\rightarrow \ldots \rightarrow$ Output):
  - Pure linear model $\rightarrow$ white box
  - Complex extensions $\rightarrow$ black box
  - Intermediate $\rightarrow$ gray boxes

7.3 Parametric vs. non-Parametric models

- All linear models (ad extensions) are called **parametric** models:
  - They make assumptions on the data distribution
  - Are bound to the amount they can learn, depending on the fixed number of parameters of the model
  - Other examples of parametric models are:
    Logistic Regression, Polynomial Regression and Perception
  - A learning model that summarizes data with a et of parameters of fixed size (independent of the number of training examples) is called a parametric model. No matter how much data you throw at a parametric model, it won't change its mind about how many parameters it needs.
- More complex models from Machine Learning are **non-parametric**:
  - Nonparametric methods are good when you have a lot of data and no prior knowledge, and when you don't want to worry too much about choosing just the right features.
  - The amount of information that the model is able to capture can grow with the increase in the amount of data analysed.
  - Examples of non-parametric models are: k-Nearest Neighbors (kNN), Decision Trees, Support Vector Machines, Neural Networks
  - They are normally more powerful (predictive power), but their interpretation is limited or basically impossible.

7.3.1 Pros and Cons (Param vs. non-Param)

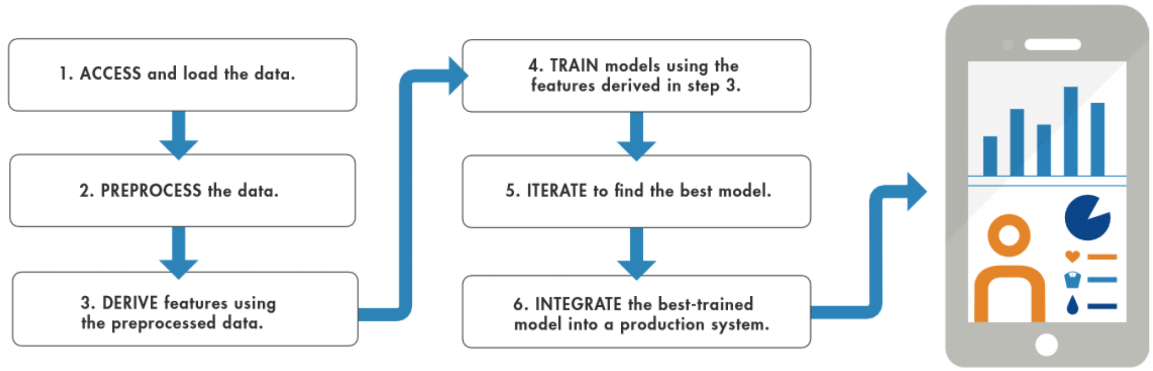
	Benefits	Limits
Param	Simpler: easier to understand and to interpret its results. Fast: faster to learn from data. Less Data: limited training data needed and can work well also with imperfect data fit.	Constrained: By choosing a functional form highly constrained to it. Limited Complexity: more suited to simpler problems. Underfitting: unlikely to match the underlying mapping function, if complex. → BIAS
Non-Param	Flexibility: Capable of fitting a large number of functional forms. Power: No assumptions (or weak assumptions) about the underlying function. Performance: Can result in higher performance models for prediction.	More data: significantly more training data required to estimate the mapping function. Slower: train require more data and then more time, due to the larger number of parameters to estimate. Overfitting: risk of overfitting the training data if not properly constrained. → VARIANCE Explicability: harder to explain why specific predictions are made, if even possible.

Rigidness (Bias) vs. Instability/Sensibility to data (Variance) tradeoff. It depends on:

- the amount of data
- the complexity and the underlying function/relationship to estimate

7.4 General approach to ML / Data Science

Mathwork guides about ML: Guide 1 and Guide 2



7.5 An example (yup, that's the title)

- This example is based on a cell phone health-monitoring app.
- The input consists of three-axial sensor data from the phone's accelerometer and gyroscope.
- The responses (or output) are the activies performed: walking, standing, running, climbing stairs or lying down.
  - **Supervised** = we have the classes
  - **Unsupervised** = we would like to discover some structure, but don't have a real output variable
- Supervised → Classification
  - The trained model (or classifier) will be integrated into an app to help users track their activity levels throughout the day.

The five steps:

- Step 1: Load the data
- Step 2: Preprocess the data
- Step 3: Derive features (predictors)
- Step 4: Build and train the model
- Step 5: Improve the model

7.5.1 Step 1: Load the Data

- To load data from the accelerometer and gyroscope, we need to do the following:
  1. Sit down holding the phone, log data from the phone, and store it in a text file labeled „Sitting“.
  2. Stand up holding the phone, log data from the phone, and store it in a second text file labeled „Standing“.
  3. Repeat the steps until we have data for each activity we want to classify.
- We store the labeled data sets in a text file.
- A flat file format such as text or CSV is easy to work with and makes it straightforward to import data.
- Machine learning algorithms are not smart enough to tell the difference between noise and valuable information.
- Before using the data for training, we need to make sure it's clean and complete.

7.5.2 Step 2: Preprocess the data

- We import the data into our environment and plot each labeled set. To preprocess the data we do the following:
  1. Look for outliers $\rightarrow$ data points that lie outside the rest of the data
     - We must decide whether the outliers can be ignored or whether they indicate a phenomenon that the model should account for.
     - In our example, they can safely be ignored (it turns out that we moved unintentionally while recording the data).
  2. Check for missing values (perhaps we lost data because the connection dropped during recording).
     - We could simply ignore the missing values, but this will reduce the size of the data set. Alternatively, we could substitute approximations for the missing values by interpolating or using comparable data from another sample (data augmentation).
  3. Remove gravitational effects from the accelerometer data so that our algorithm will focus on the movement of the subject, not the movement of the phone. A simple high-pass filter such as a biquad filter is commonly used for this.
  4. Create a training and test set (Also eventually a train-test-validation, e.g. 70% / 20% / 10%)

7.5.3 Step 3: Derive features (predictors)

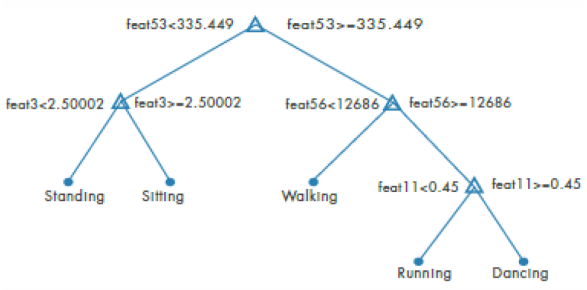
- Deriving features turns raw data into information that a machine learning algorithm can use to learn the model.
- For the activity tracker, we want to extract features that capture the frequency content of the accelerometer data. These features will help the algorithm distinguish between walking (low frequency) and running (high frequency). We create a new table that includes the selected features.
- Derive as many features as you can think of.

Data Type	Feature Selection Task	Techniques
Sensor data	Extract signal properties from raw sensor data to create higher-level information	**Peak analysis** – perform an fft and identify dominant frequencies **Pulse and transition metrics** – derive signal characteristics such as rise time, fall time, and settling time **Spectral measurements** – plot signal power, bandwidth, mean frequency, and median frequency
Image and video data	Extract features such as edge locations, resolution, and color	**Bag of visual words** – create a histogram of local image features, such as edges, corners, and blobs **Histogram of oriented gradients (HOG)** – create a histogram of local gradient directions **Minimum eigenvalue algorithm** – detect corner locations in images **Edge detection** – identify points where the degree of brightness changes sharply
Transactional data	Calculate derived values that enhance the information in the data	**Timestamp decomposition** – break timestamps down into components such as day and month **Aggregate value calculation** – create higher-level features such as the total number of times a particular event occurred

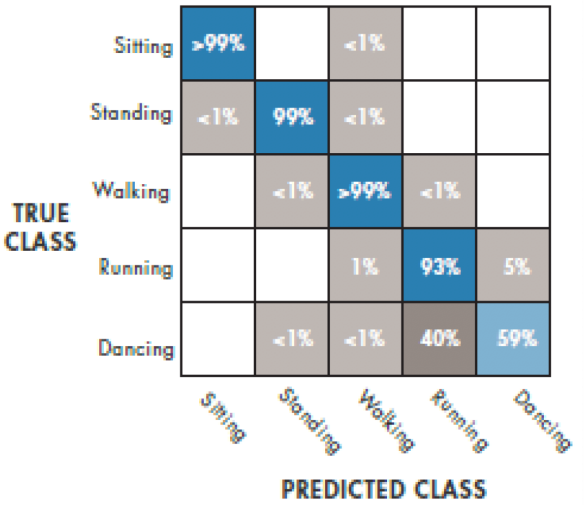
Abbildung 9: Common techniques used to derive features from data set

7.5.4 Step 4: Build and Train the Model

When building a model, it's a good idea to start with something simple; It will be faster to run and easier to interpret. We start with a basic decision tree.

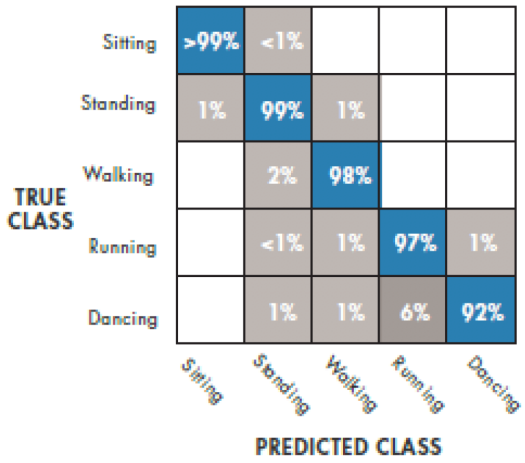


To see how well it performs, we plot the confusion matrix:

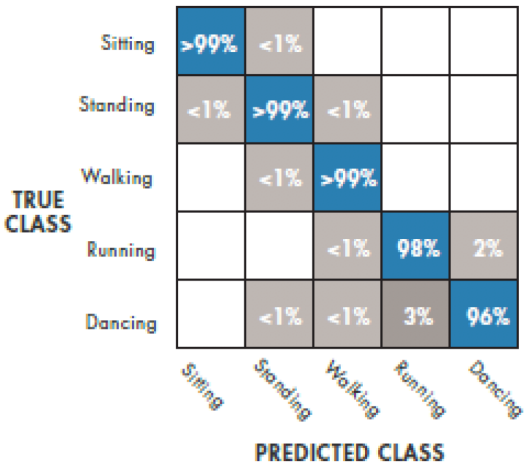


It gives us a 94.1% correct classification rate on train. Anyway, it also shows that our model is clearly having trouble distiguishing between dancing and running. Maybe a decision tree doesn't work for this type of data. Then, we will try a few different algorithms.

K-nearest neighbors (KNN), a simple algorithm that compares new points to the training data and returns the most frequent class of the „K“ nearest points. That gives us 98% accuracy compared to 94.1% for the simple decision tree and a better confusion matrix.



However, KNNs take a considerable amount of memory to run, since they require all training data to make a prediction. We try a linear discriminant model, but that doesn't improve the results. Finally, we try a multi-class support vector machine (SVM). The SVM does very well - we now get 99% accuracy.



We achieved our goal by iterating on the model and trying different algorithms. If our classifier still couldn't reliably differentiate between dancing and running, we'd look into ways to improve the model.

7.5.5 Step 5: Improve the model

Improving a model can take two different directions: make the model simpler (simplify) or add complexity. First, we look for opportunities to reduce the number of features. Popular feature reduction techniques include:

- **Correlation matrix** - shows the relationship between variables, so that variables (or features), that are not highly correlated with the output, can be removed.
- **Principal component analysis (PCA)** - eliminates redundancy by finding a combination of features that captures key distinctions between the original features and brings out strong patterns in the dataset.
- **Sequential feature reduction** - reduces features iteratively on the model until there is no improvement in performance.

Next, we look at way to reduce the model itself. We can do this by:

- Pruning branches from a decision tree
- Removing learners from an ensemble

The other approach is to add Complexity:

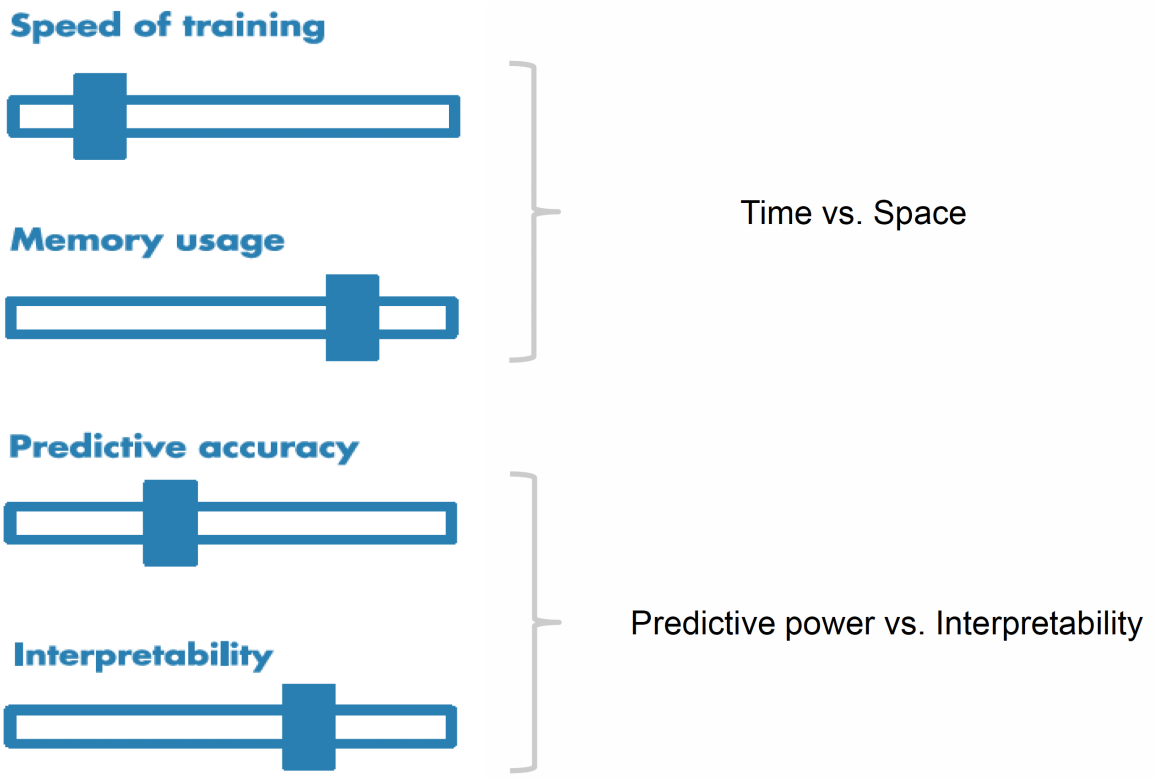
- If our model can't differentiate dancing from running because it is over-generalizing, then we need to find ways to make it more fine-tuned.

To do this, we can either:

- **Use model combination** — merge multiple simpler models into a larger model that is better able to represent the trends in the data than any of the simpler models could on their own.
- **Add more data sources** — look at the gyroscope data as well as the accelerometer data. The gyroscope records the orientation of the cell phone during activity. This data might provide unique signatures for the different activities; e.g. there might be a combination of acceleration and rotation that's unique for running.

Once we've adjusted the model, we validate its performance on the test data that we set aside during preprocessing. If the model can reliably classify activities on the test data, we're ready to move it to the phone and start tracking.

7.6 How to choose the right model (ML view)



8 Next...